Вестник Донского государственного технического университета. 2018. Т. 18, № 4. С. 449–454. ISSN 1992-5980 eISSN 1992-6006
Vestnik of Don State Technical University. 2018. Vol. 18, no. 4, pp. 449–454. ISSN 1992-5980 eISSN 1992-6006



УДК 004.451.54 https://doi.org/10.23947/1992-5980-2018-18-4-449-454

Параллельное построение двоичного дерева на основе сортировки[*]

Я. Е. Ромм[1], Д. А. Чабанюк[2]**
[1, 2] Таганрогский институт имени А. П. Чехова (филиал) Ростовского государственного экономического университета (РИНХ)", г. Таганрог, Российская Федерация

Parallel construction of binary tree based on sorting[***]

Ya. E. Romm[1], D. A. Chabanyuk[2]**
[1, 2] Taganrog Chekhov Institute, Rostov State University of Economics (RINH) branch, Rostov-on-Don, Russian Federation

Введение. Разработаны алгоритмы параллельного построения двоичного дерева. Алгоритмы выполнены на основе сортировки и описаны в конструктивной форме. Для множества из N элементов временная сложность имеет оценки $T(R) = O(1)$ и $T(R) = O(\log_2 N)$, где число процессоров $R = (N^2 - N)/2$. Дерево строится со свойством единственности. Алгоритмы инвариантны относительно вида входной последовательности. Целью работы являлась разработка и исследование способов ускорения процесса организации и преобразований древовидных структур данных на основе алгоритмов устойчивой максимально параллельной сортировки для их применения к базовым операциям информационного поиска в базах данных.
Материалы и методы. Взаимно однозначное соответствие множества входных элементов и построенного для него двоичного дерева устанавливается при помощи устойчивой адресной сортировки. Сортировка обладает максимальным параллелизмом, в операторной форме устанавливает взаимно однозначное соответствие входных и выходных индексов. На этой основе разрабатываются методы взаимного преобразования двоичных структур данных.
Результаты исследования. Получен эффективный параллельный алгоритм построения двоичного дерева на основе адресной сортировки с временной сложностью $T(N^2) = O(\log_2 N)$. От известных аналогов алгоритм отличается структурой и логарифмической оценкой временной сложности, позволяющей достигать ускорения аналогов порядка $O(N^\alpha)$, $\alpha \geq 1$. В качестве усовершенствованного варианта предложена модификация алгоритма, обеспечивающая максимально параллельное построение двоичного дерева на основе устойчивой адресной сортировки и априорного вычисления хранимых индексов корней поддеревьев. Алгоритм отличается структурой и оценкой временной сложности $T(1) = O(1)$. Аналогичная оценка достигается в последовательном варианте модифициро-

Introduction. Algorithms for the parallel binary tree construction are developed. The algorithms are based on sorting and described in a constructive form. For the N element set, the time complexity has $T(R) = O(1)$ and $T(R) = O(\log_2 N)$ estimates, where $R = (N^2 - N)/2$ is the number of processors. The tree is built with the uniqueness property. The algorithms are invariant with respect to the input sequence type. The work objective is to develop and study ways of accelerating the process of organizing and transforming the tree-like data structures on the basis of the stable maximum parallel sorting algorithms for their application to the basic operations of information retrieval on databases.
Materials and Methods. A one-to-one relation between the input element set and the binary tree built for it is established using a stable address sorting. The sorting provides maximum concurrency, and, in an operator form, establishes a one-to-one mapping of input and output indices. On this basis, methods for the mutual transformation of the binary data structures are being developed.
Research Results. An efficient parallel algorithm for constructing a binary tree based on the address sorting with time complexity of $T(N^2) = O(\log_2 N)$ is obtained. From the well-known analogues, the algorithm differs in structure and logarithmic estimation of time complexity, which makes it possible to achieve the acceleration of $O(N^\alpha)$, $\alpha \geq 1$ order analogues. As an advanced version, an algorithm modification, which provides the maximum parallel construction of the binary tree based on a stable address sorting and a priori calculation of the stored subtree root indices is suggested. The algorithm differs in structure and estimation of $T(1) = O(1)$ time complexity. A similar estimate is achieved in a sequential

[*] Работа выполнена в рамках инициативной НИР.
[**] E-mail: romm@list.ru, denchabanyuk@gmail.com
[***] The research is done within the frame of independent R&D.

ванного алгоритма, что позволяет достигать ускорения известных аналогов порядка $O(N^{\alpha})$, $\alpha > 1$.
Обсуждения и заключение. Полученные результаты направлены на организацию эффективных способов динамической обработки баз данных. Предложенные способы и алгоритмы могут составить алгоритмическую основу для ускоренного детерминированного поиска в реляционных базах данных и информационных системах.

version of the modified algorithm, which allows obtaining the acceleration of known analogs of $O(N^{\alpha})$ $\alpha > 1$ order.
Discussion and Conclusions. The results obtained are focused on the creation of effective methods for the dynamic database processing. The proposed methods and algorithms can form an algorithmic basis for an advanced deterministic search on the relational databases and information systems.

Ключевые слова: структуры данных, алгоритмы обработки данных, двоичное дерево, алгоритмы параллельной сортировки.

Keywords: data structures, data processing algorithms, binary tree, algorithms for parallel sorting.

Образец для цитирования: Ромм, Я. Е. Параллельное построение двоичного дерева на основе сортировки / Я. Е. Ромм, Д. А. Чабанюк // Вестник Дон. гос. техн. ун-та. — 2018. — Т. 18, № 4. — С. 449-454. https://doi.org/10.23947/1992-5980-2018-18-4-449-454

For citation: Ya.E. Romm, D.A. Chabanyuk. Parallel construction of binary tree based on sorting. Vestnik of DSTU, 2018, vol. 18, no.4, pp. 449-454. https://doi.org/10.23947/1992-5980-2018-18-4-449-454

Введение. В области современных высокопроизводительных вычислений имеется тенденция к конвергенции технологий параллельной обработки информации и различных архитектур процессоров. Несмотря на многообразие архитектур процессоров и способов представления информации, для повышения скорости обработки данных одной из важнейших задач информатики является идея параллельной обработки. Для ускорения обработки данных авторы предлагают использовать алгоритм устойчивой адресной сортировки, обладающей максимальным параллелизмом.

Метод параллельного построения двоичного дерева. Для массива $A = \left(a_0, a_1, \ldots, a_{n-1}\right)$ матрица сравнений строится в соответствии с [1, 2]. Элемент a_{ij} этой матрицы определяется как

$$a_{ij} = \operatorname{sign}\left(a_j - a_i\right) = \begin{cases} +, & a_j > a_i \\ 0, & a_j = a_i \\ -, & a_j < a_i \end{cases},$$

где $i, j = 1, 2, \ldots, n$.

Элемент a_j в отсортированном массиве $C = \left(c_0, c_1, \ldots, c_{n-1}\right)$ получает номер $k = \sum_{i=1}^{n} a_{ij}$, где $a_{ij} \geq 0$ при $i \leq j$, $a_{ij} > 0$ при $i > j$. Все сравнения взаимно независимы, сортировка устойчива и максимально параллельна с оценкой временной сложности $T\left(\frac{N^2 - N}{2}\right) = O(1)$. На этой основе можно выполнить параллельное построение двоичного дерева [3, 4]. Пусть дано множество из N элементов X_i, все элементы которого представлены в виде одномерного массива. На множестве предполагается заданным отношение порядка $\leq$. Требуется преобразовать массив в двоичное дерево. Для этого выполняется описанная сортировка массива. Серединный элемент массива C имеет индекс $j_{\text{ср}} = \left\lceil \frac{N}{2} \right\rceil$ и принимается за корень дерева [3]. Все элементы массива C слева от $C_{j_{\text{ср}}}$ образуют левое поддерево (левый подмассив). Элементы справа от $C_{j_{\text{ср}}}$ образуют правое поддерево (правый подмассив). Левый подмассив интерпретируется как новый массив. В нем аналогично находится индекс корня $j_{\text{ср. лев. }1/2} = \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1\right) \right\rceil = \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil$. При этом $C_{j_{\text{ср. лев. }1/2}}$ — ближайший слева потомок корня дерева $C_{j_{\text{ср}}}$. Все элементы подмассива слева от $C_{j_{\text{ср. лев. }1/2}}$ не превосходят $C_{j_{\text{ср. лев. }1/2}}$, все элементы подмассива справа не меньше $C_{j_{\text{ср. лев. }1/2}}$. Одновременно определяется индекс корня правого подмассива

$j_{\text{ср. прав. } 1/2} = \left\lceil \frac{N}{2} \right\rceil + \left\lceil \frac{1}{2}\left(\left\lceil \frac{N}{2} \right\rceil - 1\right) \right\rceil = j_{\text{ср}} + \left\lceil \frac{j_{\text{ср}} - 1}{2} \right\rceil$. При этом $C_{j_{\text{ср. прав. } 1/2}}$ — ближайший справа потомок корня дерева $C_{j_{\text{ср}}}$. Процесс рекуррентно возобновляется в каждой паре прилегающих подмассивов:

$$j_{\text{ср. лев. } 1/2^i,\, 1} = \left\lceil \frac{j_{\text{ср. лев. } 1/2^{i-1}} - 1}{2} \right\rceil,$$

$$j_{\text{ср. лев. } 1/2^i,\, 2} = j_{\text{ср. лев. } 1/2^{i-1}} + \left\lceil \frac{j_{\text{ср. лев. } 1/2^{i-1}} - 1}{2} \right\rceil,$$

$$j_{\text{ср. прав. } 1/2^i,\, 1} = j_{\text{ср. прав. } 1/2^{i-1}} - \left\lceil \frac{j_{\text{ср. прав. } 1/2^{i-1}} - j_{\text{ср. прав. } 1/2^{i-2}} - 1}{2} \right\rceil,$$

$$j_{\text{ср. прав. } 1/2^i,\, 2} = j_{\text{ср. прав. } 1/2^{i-1}} + \left\lceil \frac{j_{\text{ср. прав. } 1/2^{i-1}} - j_{\text{ср. прав. } 1/2^{i-2}} - 1}{2} \right\rceil,\ i = 1,\, 2,\, \ldots,\, \log_2 N\,.$$

В результате за время $O(1)$ формируются все элементы нижестоящего уровня двоичного дерева. Процесс можно продолжать до исчерпания $\log_2 N$ уровней двоичного дерева.

Число шагов алгоритма построения двоичного дерева в параллельной форме складывается из шага сортировки и последовательности шагов при расчете индексов корней поддеревьев. Отсюда $T(R) = \log_2 N\,\tilde{\tau} + \tau = O(\log_2 N)$, где R — число процессорных элементов, τ — время бинарного сравнения, $\tilde{\tau}$ — время вычисления одного индекса корня. Число процессоров R определяется максимально параллельной сортировкой N входных элементов, а затем вычислением индексов с удвоением по числу уровней дерева. При вычислении индексов это число не превзойдет $2^{\log_2 N - 1} = N/2$, поэтому числа процессоров, задействованных сортировкой, достаточно. В итоге R не превзойдет $\frac{N^2 - N}{2}$ [3]. Окончательно, временная сложность параллельного алгоритма построения двоичного дерева составит

$$T\left(\frac{N^2 - N}{2}\right) = O(\log_2 N).$$

Пример [3]. Двоичное дерево для массива из 15 элементов $X = (14,\ 9,\ 24,\ 7,\ 11,\ 20,\ 28,\ 3,\ 8,\ 10,\ 13,\ 17,\ 21,\ 25,\ 30)$ строится следующим образом.

Результатом сортировки является массив

$$C = (3,\ 7,\ 8,\ 9,\ 10,\ 11,\ 13,\ 14,\ 17,\ 20,\ 21,\ 24,\ 25,\ 28,\ 30).$$

Корнем двоичного дерева является серединный элемент массива C: $j_{\text{ср}} = \left\lceil \frac{15}{2} \right\rceil = 8$, $C_8 = 14$. Левый подмассив имеет корень $j_{\text{ср. лев. } 1/2} = \left\lceil \frac{8-1}{2} \right\rceil = 4$, элемент $C_4 = 9$ — корень левого поддерева, который является ближайшим слева потомком серединного элемента $C_{j_{\text{ср}}}$. Правый подмассив имеет корень $j_{\text{ср. прав. } 1/2} = 8 + \left\lceil \frac{8-1}{2} \right\rceil = 12$, элемент $C_{12} = 24$ является корнем правого поддерева и ближайшим справа потомком корня $C_{j_{\text{ср}}}$. Далее, $j_{\text{ср. лев. } 1/4,1} = \left\lceil \frac{4-1}{2} \right\rceil = 2$, элемент $C_2 = 7$ — корень поддерева слева и является ближайшим слева потомком корня поддерева $C_{j_{\text{ср. лев. } 1/2}}$. В правом поддереве корень имеет номер $j_{\text{ср. лев. } 1/4,2} = 4 + \left\lceil \frac{4-1}{2} \right\rceil = 6$, элемент $C_6 = 11$ — корень правого поддерева и ближайший справа потомок

$C_{j_{\text{ср. лев. }1/2}}$. Аналогично, слева от $C_{j_{\text{ср. прав. }1/2}}$ определяется корень $j_{\text{ср. прав. }1/4,1}=12-\left\lceil\frac{12-8-1}{2}\right\rceil=10$, элемент $C_{10}=20$ — корень левого от него поддерева и ближайший слева потомок корня поддерева $C_{j_{\text{ср. прав. }1/2}}$. Для смежного с рассмотренным правым подмассивом корень имеет номер $j_{\text{ср. прав. }1/4,2}=12+\left\lceil\frac{12-8-1}{2}\right\rceil=14$, элемент $C_{14}=28$ — ближайший справа потомок $C_{j_{\text{ср. прав. }1/2}}$ и корень правого поддерева. Нижний уровень дерева сформируют потомки, оставшиеся слева и справа от каждого из 4 идентифицированных корней (рис. 1):

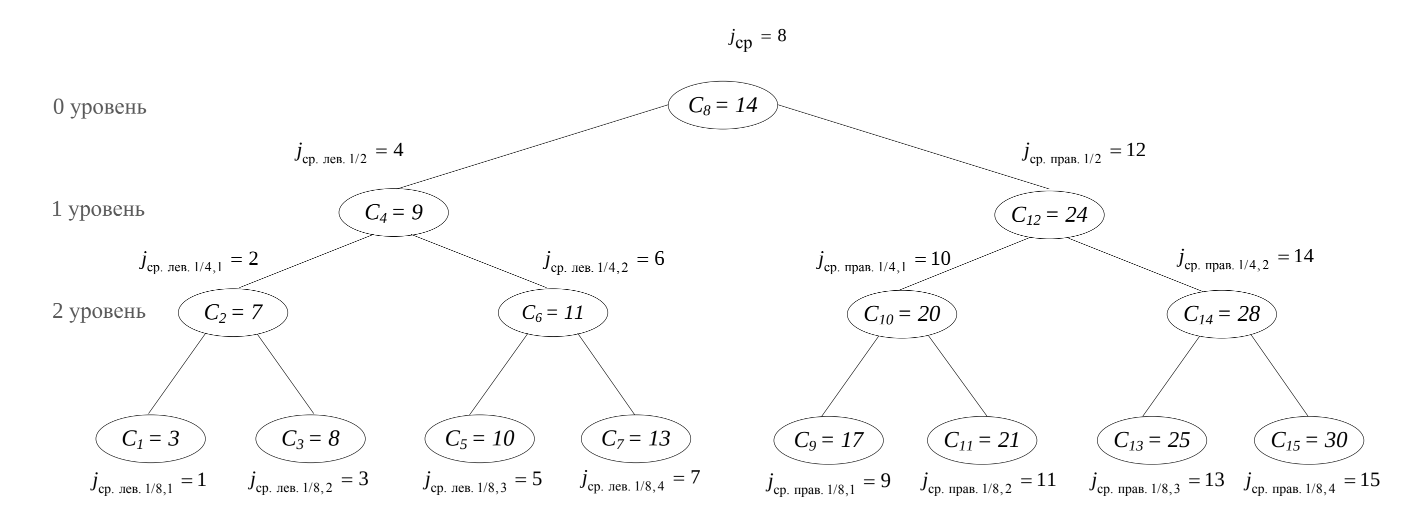


Рис. 1. Пример построения двоичного дерева на основе сортировки

Имеет место

Теорема 1 [3]. Для одномерного массива из N элементов двоичное дерево может быть построено параллельно при помощи сортировки с временной сложностью $T\left(\frac{N^2}{2}\right)=O(\log_2 N)$.

Использованная сортировка устойчива, как следствие двоичное дерево строится с единственностью. Индексы всех серединных элементов (всех корней поддеревьев) можно идентифицировать [3]. С учетом этой модификации все индексы из предыдущего примера для N значений поддеревьев можно вычислить синхронно и взаимно независимо. Это приводит к единичной оценке времени построения двоичного дерева. Для каждого конкретного N все значения индексов узлов дерева можно вычислить априори и хранить в памяти компьютера. С их помощью отсортированные элементы можно синхронно и взаимно независимо адресовать по всем узлам дерева. Формулы вычисления индексов узлов зависят только от общего количества N входных элементов и никак не зависят от их взаимного расположения после устойчивой сортировки. Для упрощения адресации памяти вычисленные индексы можно упорядочить на каждом уровне и расположить по возрастанию уровней. Тогда по ключу N считывается вся совокупность упорядоченных индексов узлов. Остается только по считанным адресам расположить отсортированные элементы дерева. На основании изложенного имеет место

Теорема 2. Для одномерного массива из N элементов двоичное дерево может быть построено параллельно при помощи сортировки и априорного вычисления индексов с временной сложностью $T\left(\frac{N^2}{2}\right)=O(1)$.

Ниже представлена единая таблица, содержащая формальные оценки временной сложности последовательных и параллельных алгоритмов построения двоичного дерева в сопоставлении с предложенными алгоритмами.

Таблица 1

Сравнительные оценки временной сложности последовательных и параллельных алгоритмов построения двоичного дерева в сопоставлении с предложенными алгоритмами

Алгоритм построения двоичного дерева	Временная сложность алгоритма	Ускорение при использовании алгоритма с единичной временной сложностью	Ускорение при использовании алгоритма с логарифмической временной сложностью
Algorithm of Lagana A., Kumar V. (2004) [5]	$\tilde{T}=O\left(N^{\frac{k+1}{k}}\right)$ [5]	$\frac{\tilde{T}}{T^*}=O\left(\frac{N^{\frac{k+1}{k}}}{1}\right)$	$\frac{\tilde{T}}{T}=O\left(\frac{N^{\frac{k+1}{k}}}{\log_2 N}\right)=O(N\ln 2)$
Algorithm of Chalermsook P. (2015) [6]	$\tilde{T}=O(N^2)$ [6]	$\frac{\tilde{T}}{T^*}=O(N^2)$	$\frac{\tilde{T}}{T}=O\left(\frac{N^2}{\log_2 N}\right)=O(N^2\ln 2)$
Полиномиальный алгоритм (2016) [7]	$\tilde{T}=O(N^3)$ [7]	$\frac{\tilde{T}}{T^*}=O(N^3)$	$\frac{\tilde{T}}{T}=O\left(\frac{N^3}{\log_2 N}\right)=O(N^3\ln 2)$
Алгоритм «left child – right sibling» (2014) [8]	$\tilde{T}=O(N^2)$ [8]	$\frac{\tilde{T}}{T^*}=O(N^2)$	$\frac{\tilde{T}}{T}=O\left(\frac{N^2}{\log_2 N}\right)=O(N^2\ln 2)$
Pattern-based algorithm (1991) [9]	$\tilde{T}=O(\lvert D\rvert\log_2 D)$ [9]	$\frac{\tilde{T}}{T^*}=O\left(\frac{\lvert D\rvert\log_2 D}{1}\right)$	$\frac{\tilde{T}}{T}=O\left(\frac{\lvert D\rvert\log_2 D}{\log_2 N}\right)$
Представленный алгоритм с логарифмической оценкой временной сложности (2015) [3]	$T=O(\log_2 N)$ [3]	–	–
Представленный алгоритм с единичной оценкой временной сложности (2015) [3]	$T^*=O(1)$	–	–

В таблице 1 D — мощность словаря шаблонов, N — число входных элементов двоичного дерева, k — размерность пространства, в котором выполняется сортировка.

Из таблицы видно, что предложенный алгоритм с логарифмической оценкой временной сложности абстрактно улучшает оценки известных алгоритмов. Минимальное ускорение достигается по отношению к алгоритму из [5]: $\frac{\tilde{T}}{T}=O(N^2\ln 2)$, или, $\frac{\tilde{T}}{T}=O(N)$, а максимальное ускорение достигается относительно полиномиального алгоритма из [7]: $\frac{\tilde{T}}{T}=O\left(\frac{N^3}{\log_2 N}\right)$ или $\frac{\tilde{T}}{T}=O(N^3)$. В случае с предложенным алгоритмом с единичной оценкой временной сложности также улучшаются оценки известных алгоритмов. В данном случае минимальное ускорение достигается относительно алгоритма из [5]: $\frac{\tilde{T}}{T^*}=O(N)$, максимальное ускорение достигается относительно полиномиального алгоритма из [7]: $\frac{\tilde{T}}{T^*}=O(N^3)$.

Заключение. Разработанные алгоритмы отличаются от известных способов [5–7, 10, 11] построения двоичного дерева тем, что используется максимально параллельная сортировка для вычисления индексов узлов. При этом для построения дерева либо затрачивается логарифмическое число шагов, либо вообще не затрачивается дополнительное время, если значения индексов априори рассчитаны для всех значений N в некоторых реальных границах и хранятся в памяти компьютера. Предложенный параллельный алгоритм построения двоичного дерева может использоваться с целью организации эффективных способов динамической обработки баз данных.

Библиографический список

1. Ромм, Я. Е. Сравнение слов с единичной временной сложностью / Я. Е. Ромм, Д. А. Чабанюк // Известия Южного федер. ун-та. Технические науки. — 2014. — № 7 (156). — С. 230–238.
2. Ромм, Я. Е. Параллельная сортировка слиянием по матрицам сравнений. II / Я. Е. Ромм // Кибернетика и системный анализ. — 1995. — № 4. — С. 13–37.
3. Ромм, Я. Е. Построение двоичного дерева на основе параллельной сортировки / Я. Е. Ромм, Д. А. Чабанюк // Фундаментальные исследования. — 2015. — Т. 8., № 3. — С. 509–513.
4. Ромм, Я. Е. Параллельное построение двоичного дерева на основе сортировки / Я. Е. Ромм, Д. А. Чабанюк // Аспекты развития науки, образования и модернизации промышленности : матер. Всеросс. научно-практ. конф. — Таганрог, 2017. — Т. 1. — С. 77–84.
5. Laganà A. Computational Science and Its Applications: Lecture Notes in Computer Science / A. Laganà, V. Kumar, C. Tan. — Assisi: Springer Science & Business Media, 2004. — 1044 p. — DOI: https://doi.org/10.1007/b98048
6. Chalermsook P. 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015) / P. Chalermsook, M. Goswami; eds. – Piscataway, NJ: IEEE, 2015. – 410-423 p. – DOI: 10.1109/FOCS.2015.98
7. Гавриков, А. В. *T*-неприводимые расширения для ориентированных бинарных деревьев / А. В. Гавриков // Компьютерные науки и информационные технологии. — 2016. — № 6. — С. 123–125. — DOI : https://doi.org/10.17223/20710410/34/6
8. Гриценко, Н. С. Построение двоичного дерева на основе модифицированной схемы хранения деревьев общего вида «*left child*»-«*right sibling*» (*LCRS*) / Н. С. Гриценко, Ю. С. Белов // Инженерный журнал : наука и инновации. — 2014. — № 3. — С. 75–84. — DOI: https://doi.org/10.18698/2308-6033-2014-3-1281.
9. Amir A. Adaptive dictionary matching / A. Amir, M. Farach // Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on. — IEEE, 1991. — P. 760-766. — DOI: https://doi.org/10.1109/SFCS.1991.185445
10. Fischer J. Theoretical and Practical Improvements on the RMQ-Problem, with Applications to LCA and LCE / J. Fischer, V. Heun // Combinatorial Pattern Matching – Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. – Vol. 4009. – P. 36-48.
11. Institute of Electrical and Electronics Engineers. Pattern-Avoiding Access in Binary Search Trees / Computer Society // 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS 2015). – 2015. – № 56. — P. 410-423. DOI: https://doi.org/10.1109/FOCS.2015.32

Поступила в редакцию 21.01.2018
Сдана в редакцию 24.01.2018
Запланирована в номер 20.07.2018

Received 21.01.2018
Submitted 24.01.2018
Scheduled in the issue 20.07.2018

Об авторах

Ромм Яков Евсеевич,
заведующий кафедрой «Информатика» Таганрогского института им. А. П. Чехова (филиал) Ростовского государственного экономического университета (РИНХ) (РФ, 347936, г. Таганрог, ул. Инициативная, д. 48), доктор технических наук, профессор,
ORCID: http://orcid.org/0000-0002-7251-2844
romm@list.ru

Чабанюк Денис Андреевич,
доцент кафедры «Теоретическая, общая физика и технологии» Таганрогского института им. А. П. Чехова (филиал) Ростовского государственного экономического университета (РИНХ) (РФ, 347936, г. Таганрог, ул. Инициативная, д. 48),
ORCID http://orcid.org/0000-0003-2972-0944
denchabanyuk@gmail.com

Authors:

Romm, Yakov Ye.,
Head of the Information Technology Department, Taganrog Chekhov Institute, Rostov State University of Economics (RINH) branch (48, Initsiativnaya St., Taganrog, RF), Dr.Sci. (Eng.), professor,
ORCID: http://orcid.org/0000-0002-7251-2844
romm@list.ru

Chabanyuk, Denis A.,
associate professor of the Theoretical, General Physics and Technologies Department, Taganrog Chekhov Institute, Rostov State University of Economics (RINH) branch (48, Initsiativnaya St., Taganrog, RF),
ORCID http://orcid.org/0000-0003-2972-0944
denchabanyuk@gmail.com